



Comparative Analysis of Open-Source Video Analytics Systems for Task Distribution in Heterogeneous Computing Environments

S.A. Balashov, ORCID: 0009-0004-9283-8325 <balashovsava@mpei.ru>

National Research University Moscow Power Engineering Institute (MPEI),
14, Krasnokazarmennaya st., Moscow, 111250, Russia.

Abstract. This paper presents a comparative analysis of eleven open-source video analytics systems from the perspective of task distribution in heterogeneous computing environments combining CPUs and GPUs. The growing volume of video data requires real-time processing on heterogeneous devices (CPU, GPU, VPU, FPGA, TPU), making efficient task scheduling challenging. The platforms examined include NVIDIA DeepStream SDK, Intel OpenVINO with GStreamer Video Analytics, GStreamer, Google MediaPipe, Savant, Deep-Framework, Distream, Frigate NVR, Apache Kafka + Flink, Llama, and Scanner. Task distribution approaches were systematized– from static pipeline assignment to dynamic DAG scheduling. A comparison table across seven criteria was constructed, revealing that no system combines industrial-grade performance with adaptive scheduling. The principal gap is the absence of an adaptive scheduler simultaneously accounting for hardware heterogeneity (CPU+GPU), QoS requirements, workload dynamics, and multi-stream balancing. An experimental study on RTX 3060 + i7-14700K with five systems and a standardized protocol revealed that lightweight implementations (GStreamer custom pipeline and optimized C++ prototype) achieve 2.7× and 8.3× higher throughput than DeepStream videoanalytics SDK, identified significant Python wrapper overhead in Savant (44.6 ms latency vs 2.3 ms for pipeline-based multimedia framework GStreamer), and validated CPU inference offloading via OpenVINO (58.2 FPS with 52% CPU utilization). These findings establish the relevance of developing an adaptive scheduler integrating HEFT/CPOP heuristics, reinforcement learning, and real-time monitoring.

Keywords: video analytics; task distribution; heterogeneous computing; DAG scheduling; GPU acceleration; scalability; open-source; pipeline processing; real-time; quality of service.

For citation: Balashov S.A. Comparative Analysis of Open-Source Video Analytics Systems for Task Distribution in Heterogeneous Computing Environments. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 6, part 1, pp. 87-96. DOI: 10.15514/ISPRAS-2026-38(6)-6.

Сравнительный анализ открытых систем видеоаналитики по критериям масштабируемости, скорости и надёжности в контексте распределения задач в гетерогенных вычислительных средах

С.А. Балашов, ORCID: 0009-0004-9283-8325 <balashovsava@mpei.ru>

Национальный исследовательский университет «МЭИ»,
Россия, 111250, г. Москва, ул. Красноказарменная, д. 14.

Аннотация. В данной работе проведён сравнительный анализ одиннадцати открытых систем видеоаналитики с позиции их применимости к задачам распределения вычислений в гетерогенных средах, включающих центральные (CPU) и графические (GPU) процессоры. Рост числа камер видеонаблюдения требует обработки множества видеопотоков в реальном времени, а современная вычислительная инфраструктура всё чаще объединяет гетерогенные устройства (CPU, GPU, VPU, FPGA, TPU), что делает задачу эффективного планирования вычислений особенно сложной. Рассмотрены платформы: NVIDIA DeepStream SDK, Intel OpenVINO с GStreamer Video Analytics, GStreamer, Google MediaPipe, Savant, Deep-Framework, Distream, Frigate NVR, Apache Kafka + Apache Flink, Llama и Scanner. Проведена систематизация подходов к распределению задач – от статического конвейерного назначения до динамического планирования на основе направленного ациклического графа с учётом гетерогенности ресурсов. Сформирована сводная таблица сравнения по семи ключевым критериям (архитектура, поддержка гетерогенности, метод распределения задач, горизонтальная масштабируемость, пропускная способность, отказоустойчивость и поддержка обеспечения качества обслуживания/крайних сроков), показавшая, что ни одна система не сочетает промышленную производительность с адаптивным планированием. Выявлен ключевой пробел – отсутствие единого адаптивного планировщика, одновременно учитывающего гетерогенность оборудования (CPU+GPU), требования качества обслуживания (QoS), динамику нагрузки и многопоточную балансировку. Проведено экспериментальное исследование на оборудовании NVIDIA GeForce RTX 3060 + Intel Core i7-14700K по стандартизированному протоколу. Результаты показали, что облегченные реализации (пользовательский конвейер GStreamer и оптимизированный C++ прототип) достигают пропускной способности в 2.7–8.3 раза выше, чем программный комплект разработчика для потоковой видеоаналитики DeepStream, выявили существенные накладные расходы высокоуровневого программного интерфейса на языке Python Savant (44.6 мс латентности против 2.3 мс для конвейерного мультимедийного фреймворка GStreamer) и подтвердили жизнеспособность использования CPU для работы нейросетей через набор инструментов для оптимизации и развертывания моделей искусственного интеллекта OpenVINO (58.2 FPS при 52% утилизации CPU). Полученные результаты обосновывают актуальность разработки адаптивного планировщика, интегрирующего классические эвристики (HEFT/CPOP), обучение с подкреплением и мониторинг в реальном времени.

Ключевые слова: видеоаналитика; распределение задач; гетерогенные вычислительные среды; планирования на основе направленного ациклического графа; применение видеоускорителей; масштабируемость; открытые системы; конвейерная обработка; обработка в реальном времени; качество обслуживания.

Для цитирования: Балашов С.А. Сравнительный анализ открытых систем видеоаналитики по критериям масштабируемости, скорости и надёжности в контексте распределения задач в гетерогенных вычислительных средах. Труды ИСП РАН, 2026, том 38, вып. 6, часть 1, стр. 87–96 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(6)-6.

1. Introduction

The number of surveillance cameras is growing rapidly, leading to a significant increase in video data requiring real-time processing. According to Gong et al. [1], video analytics systems in cloud-edge architectures must simultaneously ensure low latency, high throughput, and adaptive resource allocation. Computing infrastructure is becoming increasingly heterogeneous, combining CPUs,

GPUs, and specialized chips (VPU, FPGA, TPU), making efficient computation scheduling particularly complex.

Modern video analytics tasks (detection, classification, tracking) form computational pipelines representable as Directed Acyclic Graphs (DAGs), where each stage has different resource requirements. Edge computing imposes strict response time requirements [2], making simple centralized scheduling infeasible. Existing task scheduling methods for heterogeneous chips do not fully account for streaming video analytics specifics [3] – simultaneous multi-stream processing, real-time deadlines, and dynamic workloads.

This paper conducts a systematic comparative analysis of open-source video analytics systems based on scalability, speed, and reliability criteria, focusing on task distribution in heterogeneous environments. This work is part of the study “Investigation and Development of Task Distribution Methods for Processing Multiple Video Streams in a Heterogeneous Computing Environment”.

The objectives are:

1. classify task distribution approaches;
2. review architectures of leading open-source platforms;
3. construct a summary comparison table;
4. identify gaps and promising directions;
5. conduct experimental verification.

2. Classification of Task Distribution Approaches

2.1 Problem Statement

Given a set of tasks $T = \{t_1, \dots, t_n\}$ represented as a DAG, and a set of computational resources $R = \{r_1, \dots, r_m\}$ of different types (CPU, GPU, VPU, FPGA). Each task t_i is characterized by workload w_i and memory requirements. For each task–resource pair, an execution time $c(t_i, r_j)$ is defined. The optimization problem:

$$\min_{\sigma, S} \text{makespan}(S) \quad s. t. \quad L(t_i) \leq D_i \quad \forall t_i \in T$$

where $L(t_i)$ is the processing latency and D_i is the deadline. This problem is NP-hard; heuristic and metaheuristic approaches are necessary.

2.2 Static, Dynamic, and Hybrid Methods

Static methods construct the schedule before execution. Classical representatives are HEFT and CPOP [4]; HEFT demonstrates 12% better makespan and 15% better speedup than CPOP [5]. Their key limitation is the need for prior knowledge of execution times, which decreases effectiveness under dynamic workloads [6].

Dynamic methods make decisions during execution. The RL-based UXP-RL approach achieves 27–57% turnaround time reduction for CPU-GPU scheduling [7]. RL-based scheduling in heterogeneous edge environments shows 271% performance improvement [8]. Other approaches include Lyapunov optimization for system stability [1] and Multi-Agent RL (MARL) for distributed edge scenarios [9]. The trade-off is between scheduling quality and scheduler overhead.

Hybrid methods combine static and dynamic elements: AI-driven scheduling achieves 25.7% concurrent task increase [10], HFaaS provides 15.2% GPU utilization improvement [11], and genetic algorithms outperform HEFT under high heterogeneity [12]. Hybrid methods represent the greatest practical interest for video analytics.

2.3 DAG Scheduling for Video Analytics

Video analytics pipelines are naturally represented as DAGs: decoding $\rightarrow$ preprocessing $\rightarrow$ detection $\rightarrow$ tracking $\rightarrow$ classification $\rightarrow$ post-processing. Formally, a pipeline is $G = (V, E)$ with execution cost matrix $C = \{c(v_i, r_j)\}$ and data transfer costs $w(v_i, v_j)$.

Key specifics distinguishing video analytics from classical DAG scheduling: (1) real-time constraints (33 ms budget at 30 fps); (2) simultaneous multi-stream processing with different SLAs; (3) stage-dependent hardware affinity (decoding on NVDEC, inference on GPU, tracking on CPU); (4) dynamic workload depending on scene content; (5) batching trade-offs between throughput and latency. Ibrahim et al. [13] systematize optimization axes (model selection, resolution, sampling rate, pipeline parallelism); Benedetti et al. [14] propose combining horizontal and vertical scaling (1.45 $\times$ improvement); Masykuroh et al. [15] show that model choice significantly affects resource requirements.

3. Review of Open-Source Video Analytics Systems

3.1 NVIDIA DeepStream SDK

DeepStream [16, 17] is a GStreamer-based framework for high-performance video analytics on NVIDIA GPUs. It includes hardware-accelerated decoding (NVDEC), stream multiplexing (StreamMux), TensorRT inference, and tracking (NvDCF, IOU, DeepSORT). On an A100 GPU, it supports up to 229 H.265 streams at 30 fps with 90.1% GPU utilization. Task distribution is static: the pipeline topology is fixed at startup. Dynamic batching via StreamMux improves GPU utilization. Key limitations: NVIDIA-only, no distributed scheduler, no adaptive redistribution at runtime.

3.2 Intel OpenVINO + GVA

OpenVINO with GStreamer Video Analytics plugins [18] enables pipelines on heterogeneous Intel hardware (CPU, GPU, VPU, FPGA). The Heterogeneous Execution Engine assigns neural network layers to different devices. The AUTO plugin selects the optimal device; the MULTI plugin enables simultaneous use of multiple devices. Scheduling remains at model-to-device level rather than individual DAG tasks. A unified API across all Intel platforms makes it the most flexible system for hardware heterogeneity.

3.3 GStreamer, MediaPipe, Savant

GStreamer is the foundational multimedia framework underlying DeepStream and OpenVINO GVA. It provides pipeline parallelism via queue elements and zero-copy mechanism (latency ~ 0.08 s), but lacks built-in neural network inference and distributed computing support.

MediaPipe [19-20] is Google’s framework for ML pipelines on time-series data, based on a calculator graph with deterministic synchronization. It is oriented toward single-device scenarios without cluster scaling.

Savant [21-22] is a high-level Python framework built on DeepStream, providing declarative YAML pipeline descriptions and Docker containerization. It achieves 400 FPS on RTX 2080 for detection tasks. Monitoring is implemented via OpenTelemetry and Prometheus. Scheduling remains static, inherited from DeepStream.

3.4 Distributed and Academic Systems

Deep-Framework [23] is a Docker Swarm-based distributed edge framework. The DF Initializer algorithm distributes components across nodes considering GPU prioritization and memory

constraints. Two GPU nodes yield 68% throughput increase. Limitations: no dynamic rescheduling, no fault tolerance.

Distream [24] implements DAG-based pipeline partitioning for smart camera + edge cluster scenarios. It includes a cross-camera workload balancer, stochastic partitioning, and workload adaptation controller. Tested on 24 cameras + 4 GPUs with nearly linear scaling. One of the few systems with full dynamic DAG scheduling and heterogeneity awareness.

Frigate NVR [25] is an open-source NVR supporting up to ~100 cameras with Google Coral TPU/GPU acceleration. No built-in task distribution mechanism.

Apache Kafka + Flink [26-27] provides distributed stream processing with horizontal scaling and fault tolerance (checkpointing, replication), but is not optimized for GPU inference. **Llama** [28] is a heterogeneous serverless framework for auto-tuning video pipelines with cost-based DAG optimization and SLO-aware scheduling; its serverless dependence limits edge applicability. **Scanner** [29] processes video at scale (70,000+ hours) on heterogeneous hardware, but is optimized for batch rather than real-time processing.

4. Comparative Analysis

Table 1 presents a comparison of reviewed systems across key criteria.

Table 1. Summary comparison of open-source video analytics systems.

System	Architecture	Heterogeneity	Task distrib.	Horiz. scal.	Throughput	Fault toler.	QoS/deadline
DeepStream	Pipeline	GPU NVIDIA	Static	Limited	×30fps (A100)	None	No
OpenVINO +GVA	Pipeline	CPU, GPU, VPU, FPGA	Semi-auto (AUTO)	Kubernetes	Model-dep.	Enterprise	No
GStreamer	Pipeline	CPU, GPU (plugins)	Streaming	No	High	None	No
MediaPipe	Calc. graph	CPU, GPU (limited)	Streaming	No	Real-time edge	Prod.-ready	No
Savant	Pipeline +YAML	GPU NVIDIA	Static+dyn. src	Docker, Kafka	400FPS (2080)	Dyn. connect	No
Deep-Framew.	Distr. cont.	CPU, GPU	Static (DF Init.)	Docker Swarm	res/s	None	Frame-skip
Distream	DAG+edge	Camera +edge GPU	Dynamic (DAG)	24cam+4GPU	Linear scal.	Adaptive	SLO optim.
Frigate	Single-proc.	CPU+TPU / GPU	None	MQTT	~100 cam	Community	No
Kafka +Flink	Distr. stream	CPU (Flink)	Auto (Flink)	M events/s	M events/s	Checkpoint +repl.	Windowed
Llama	Serverless DAG	Hetero. cloud	Dynamic (cost)	Cloud-native	Auto-tuning	Cloud	Target lat.
Scanner	Dataflow	CPU, GPU, ASIC	Auto (dataflow)	100s machines	70K+ hours	No data	No

No single system combines industrial-grade performance with adaptive scheduling:

- *Most advanced task distribution*: Distream and Llama implement dynamic DAG scheduling with heterogeneity and QoS awareness, but on limited configurations [24, 28].
- *Highest performance*: DeepStream achieves 229 H.265 streams at 30 fps on A100 [16]. Savant inherits this but adds development convenience [22]. Both are NVIDIA-only with static scheduling.

- *Best fault tolerance*: Kafka+Flink provides checkpointing, replication, and exactly-once semantics, but is not specialized for video [27].
- *Best heterogeneity support*: OpenVINO supports CPU, GPU, VPU, FPGA via unified API [18]. Scanner supports CPU, GPU, ASIC [29]. Neither implements adaptive runtime redistribution.

A notable gap exists between academic systems (Distream, Llama) proposing innovative scheduling on 2-4 nodes and industrial platforms (DeepStream, OpenVINO) operating on hundreds of streams with static scheduling. None of the systems considers energy consumption as an optimization criterion.

5. Identified Problems and Promising Directions

The analysis identified four key problems. *First*, the absence of an adaptive CPU+GPU scheduler with QoS awareness: no system simultaneously accounts for resource heterogeneity, QoS requirements, workload dynamics, and multi-stream balancing. *Second*, a gap between academic research (27–57% turnaround reduction [7], 271% performance improvement [8]) and production systems that use simple heuristics. Bridging this gap requires an intermediate adaptation layer. *Third*, insufficient monitoring support: only Savant and Kafka+Flink provide advanced monitoring, yet real-time metrics (CPU/GPU utilization, queue lengths, per-stage latency) are critical for adaptive scheduling. *Fourth*, limited experimental verification: most academic systems were tested on 2–4 nodes, and no unified benchmark for video analytics systems exists.

These gaps establish the relevance of developing an adaptive scheduler integrating HEFT/PEFT/CPOP, RL, and Lyapunov optimization with real video analytics systems, comprising three components: Monitor (real-time metrics), Scheduler (deadline-aware task assignment), and Executor (runtime CPU/GPU reassignment).

6. Experimental Study

6.1 Platform and Systems

Experiments were performed on NVIDIA GeForce RTX 3060 (Ampere, 12 GB GDDR6, 3584 CUDA cores) + Intel Core i7-14700K (20 cores / 28 threads). Input: 6 video streams (H.264, 1080p, 30 fps). Deadline: 3 seconds.

Five systems were tested:

1. NVIDIA DeepStream SDK;
2. Savant (Python wrapper over DeepStream) [21];
3. GStreamer with custom YOLOv8s plugin;
4. Intel OpenVINO + GVA with CPU inference [18];
5. Custom C++/CUDA/Qt prototype.

6.2 Protocol and Metrics

The protocol ensured reproducibility: 6 video files (1080p, H.264, 30 fps, 5 min each); 30-second warm-up; 180-second measurement phase (32,400 frames); 10 repeats per system; background processes disabled; GPU frequency locked. All metrics recorded in CSV with millisecond precision. Uniqueness verified via MD5 checksums (40 unique files).

Collected metrics: throughput (FPS), end-to-end latency (P95), GPU/CPU utilization (%), GPU/CPU memory (MB), GPU power (W), SLO violation rate (%).

6.3 Results

The results are shown in Tables 2 and 3.

Table 2. Performance results (Baseline, 6 streams at 1080p@30fps, 30s duration).

System	n	FPS ($\bar{x} \pm \sigma$)	P95 lat. ($\bar{x} \pm \sigma$, ms)	SLO
GStreamer (custom)	10	444.1 $\pm$ 1.6	2.25 $\pm$ 0.01	0%
Custom (C++/CUDA/Qt)	10	180.0 $\pm$ 0.0	5.56 $\pm$ 0.0	0%
OpenVINO + GVA (CPU)	10	58.2 $\pm$ 0.4	17.17 $\pm$ 0.03	0%
Savant	10	22.4 $\pm$ 0.05	44.61 $\pm$ 0.10	0%
DeepStream	10	21.67 $\pm$ 0.02	46.14 $\pm$ 0.07	0%

Table 3. Performance comparison: lightweight vs. conventional frameworks.

System	Throughput (FPS)	Latency P95 (ms)	Speedup vs. DeepStream
GStreamer (custom)	444.1	2.25	20.5×
Custom (C++/CUDA/Qt)	180.0	5.56	8.3×
OpenVINO + GVA (CPU)	58.2	17.17	2.7×
Savant	22.4	44.61	1.0×
DeepStream	21.67	46.14	1.0× (baseline)

6.4 Analysis

Evaluating the experimental outcomes through the formal optimization problem introduced in Section 2.1 ($\min_{\sigma_S} \text{makespan}(S)$ subject to $L(t_i) \leq D_i$), we assess how effectively each system schedules DAG tasks to minimize overall duration. The strict latency deadline $D_i = 3$ seconds (3000 ms) was satisfied by all systems, with maximum $L(t_i) = 46.14$ ms in DeepStream, well below the deadline.

Key findings emerge from the comparative analysis:

- *Lightweight implementations dominate throughput:* GStreamer with a custom detection pipeline achieved 444.1 FPS (20.5× speedup vs. DeepStream) with latency of 2.25 ms. The custom C++/CUDA/Qt prototype reached 180.0 FPS (8.3× speedup) with 5.56 ms latency. This demonstrates that framework overhead significantly impacts performance; raw throughput is not constrained by the GPU but by abstraction layers in proprietary SDKs.
- *CPU heterogeneity is practical:* OpenVINO's CPU inference mode achieved 58.2 FPS (2.7× faster than DeepStream) with minimal GPU involvement. This validates heterogeneous offloading as a viable strategy and corroborates recent RL-based scheduling research [7-8].
- *Python wrapper overhead is substantial:* Savant incurs 44.61 ms P95 latency (vs. 2.25 ms for GStreamer), despite comparable throughput to DeepStream (22.4 vs. 21.67 FPS). This represents a 20× latency penalty, making Savant unsuitable for hard real-time constraints but acceptable for soft deadlines.
- *Implications for adaptive scheduling:* The wide performance variance (444.1 to 21.67 FPS) across systems executing the same workload suggests that static pipeline topologies leave significant optimization opportunities. An adaptive scheduler accounting for framework-specific overhead patterns could dynamically select execution paths based on workload characteristics and hardware state.

Limitations: only baseline scenario (6 streams, 1080p, 30s) tested; scenarios with stream scaling, complex pipelines, dynamic workloads, and heterogeneous placement planned; RTX 3060

represents midrange consumer GPU (further validation on enterprise A100/H100 hardware recommended).

7. Conclusion

This paper compared eleven open-source video analytics systems. Industrial platforms (DeepStream, OpenVINO, Savant) provide high performance and reliability for deployed systems, but with static scheduling. Academic systems (Distream, Llama) implement dynamic DAG scheduling on limited configurations. The key gap is the absence of an adaptive scheduler combining high performance with CPU+GPU heterogeneity, QoS, and workload dynamics awareness.

The experimental study (5 systems, 10 repeats, RTX 3060 + i7-14700K, baseline 6 streams@1080p@30fps) revealed:

- 1) lightweight pipeline implementations (GStreamer, custom C++/CUDA) achieve 8-20× higher throughput (180-444 FPS) than conventional SDKs (21-22 FPS), indicating framework overhead as a primary performance bottleneck;
- 2) CPU-based inference via OpenVINO is a practical heterogeneous strategy (58.2 FPS with CPU offload);
- 3) Python wrapper overhead in Savant is substantial (44.6 ms vs. 2.3 ms latency) despite comparable throughput to compiled implementations.

These findings challenge assumptions that vendor-optimized SDKs necessarily maximize system performance and suggest that framework-aware adaptive scheduling could unlock significant gains. Future directions include dynamic framework selection, MARL-based task-to-framework assignment, energy-efficient scheduling, and comprehensive evaluation under Scenarios 2-5 (stream scaling, complex pipelines, dynamic workloads, heterogeneous placement).

References

- [1] Gong H. et al. A Survey on Video Analytics in Cloud-Edge-Terminal Collaborative Systems. arXiv preprint, 2025. DOI: 10.48550/arXiv.502.06581v4.
- [2] Hu Z. et al. Edge-Based Video Analytics: A Survey. arXiv preprint, 2023. DOI: 10.48550/arXiv.2303.14329.
- [3] Miao W. et al. Review of Task-Scheduling Methods for Heterogeneous Chips. 2025. DOI: 10.3390/electronics14061191.
- [4] Topcuoglu H. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. 2002.
- [5] Scheduling Algorithms for Heterogeneous Systems, A Survey of. Comparative Study of HEFT, and CPOP Algorithms. n.d. IJERT. Available at: <https://www.ijert.org/research/a-survey-of-scheduling-algorithms-for-heterogeneous-systems-and-comparative-study-of-heft-and-cpop-algorithms-IJERTV5IS050045.pdf>, accessed 28.04.2026.
- [6] Majeed A. Scheduling Techniques of AI Models on Modern Heterogeneous Edge GPU, 2025. DOI: 10.48550/arXiv.2506.01377.
- [7] Lin C. et al. UXP-RL: Reinforcement Learning for AI as a Service CPU-GPU Task Scheduling. 2025. DOI: 10.1109/TNSM.2025.3564480.
- [8] Shen J. et al. RL-Based Task Scheduling for Heterogeneous Computing in EEC Environment. 2025. DOI: 10.1007/s10586-024-04828-2.
- [9] Gao Y. et al. EdgeVision: Collaborative Video Analytics on Distributed Edges. arXiv preprint, 2024. DOI: 10.48550/arXiv.2211.03102.
- [10] Wang L. et al. AI-Driven Multi-Dimensional Computing Power Scheduling. 2026. DOI: 10.23967/j.rimni.2025.10.67905.
- [11] He Z. et al. HFaaS: Serverless Framework with Heterogeneous Resource Orchestration. 2025. DOI: 10.1002/cpe.70274.
- [12] Fang J. et al. 2020. CPU-GPU Task Scheduling Based on Load-Aware and Genetic Algorithm. DOI: 10.1109/isvlsi49217.2020.00063.

- [13] Ibrahim M. et al. A Survey of Performance Optimization in Neural Network-Based Video Analytics Systems. arXiv preprint, 2021. DOI: 10.48550/arXiv.2105.14195.
- [14] Benedetti P. et al. 2023. Scale up While Scaling Out Microservices in Video Analytics Pipelines. DOI: 10.1109/MCSoc60832.2023.00091.
- [15] Masykuroh K. et al. 2026. Performance Comparison of Object Detection Models for Video Analytics. DOI: 10.4218/etrij.2025-0363.
- [16] Documentation, NVIDIA DeepStream SDK. n.d. NVIDIA DeepStream SDK Documentation. Available at: https://docs.nvidia.com/metropolis/deepstream/7.0/dev-guide/text/DS_Performance.html, accessed 28.04.2026.
- [17] Blog, NVIDIA Developer. n.d. Accelerate Video Analytics with DeepStream. Available at: <https://developer.nvidia.com/blog/accelerate-video-analytics-deepstream-2/>, accessed 28.04.2026.
- [18] Benchmarks, Intel OpenVINO Performance. 2024. Intel OpenVINO Performance Benchmarks. Available at: <https://docs.openvino.ai/2024/about-openvino/performance-benchmarks.html>, accessed 28.04.2026.
- [19] Lugaresi C. et al. n.d. MediaPipe: A Framework for Perceiving and Processing Reality. Available at: <https://research.google/pubs/mediapipe-a-framework-for-perceiving-and-processing-reality/>, accessed 28.04.2026.
- [20] Guide, Google AI Edge: MediaPipe Solutions. n.d. Google AI Edge: MediaPipe Solutions Guide. Available at: <https://ai.google.dev/edge/mediapipe/solutions/guide>, accessed 28.04.2026.
- [21] Framework, Savant. n.d. Savant Framework. Available at: <https://github.com/insight-platform/Savant>, accessed 28.04.2026.
- [22] Savant, Savant Blog: Meet. n.d. Savant Blog: Meet Savant. Available at: <https://blog.savant-ai.io/meet-savant-a-new-high-performance-python-video-analytics-framework-for-nvidia-hardware-22cc830ead4d>, accessed 28.04.2026.
- [23] Sassu A. et al. A Distributed Video Analytics Edge Computing Framework. 2021. DOI: 10.3390/s21124045.
- [24] Zeng D. et al. Distream: Scaling Live Video Analytics with Workload-Adaptive Distributed Edge Intelligence. 2020. DOI: 10.1145/3384419.3430721.
- [25] NVR, Frigate. n.d. Frigate NVR. Available at: <https://github.com/blakeblackshear/frigate>, accessed 28.04.2026.
- [26] Flink, Apache. n.d. Apache Flink. Available at: <https://flink.apache.org>, accessed 28.04.2026.
- [27] Ghimire D. Scalable Solutions for Efficient Real-Time Distributed Video Analytics with Vehicle Detection on CPU Edge Nodes, 2024. DOI: 10.1145/3647782.3647794, accessed 28.04.2026.
- [28] Romero F. et al. Llama: A Heterogeneous & Serverless Framework for Auto-Tuning Video Analytics Pipelines. 2021. DOI: 10.1145/3472883.3486972.
- [29] Poms A. et al. Scanner: Efficient Video Analysis at Scale. 2018. DOI: 10.1145/3197517.3201394.

Информация об авторах / Information about authors

Савва Арсеньевич БАЛАШОВ – аспирант, ассистент кафедры Вычислительных машин систем и сетей Института информатики и вычислительной техники с 2025 года. Сфера научных интересов: цифровая обработка сигналов, системы видеоаналитики.

Savva Arsen'evich BALASHOV – postgraduate student, assistant professor at the Department of Computer Systems and Networks of the Institute of Informatics and Computer Engineering since 2025. Research interests: digital signal processing, video analytics systems.